ARMY RESEARCH LABORATORY

Dynamically Generated Nodes and Links for a Dynamic Network Structure Using X3D

by Andrew M. Neiderer

ARL-MR-719 **May 2009**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-MR-719 **May 2009**

Dynamically Generated Nodes and Links for a Dynamic Network Structure Using X3D

Andrew M. Neiderer
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
May 2009	Final	August 2008–January 2009

4. TITLE AND SUBTITLE
Dynamically Generated Nodes and Links for a Dynamic Network Structure Using X3D

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Andrew M. Neiderer

5d. PROJECT NUMBER
9TEPTC

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-CI-IC
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-MR-719

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report presents an Extensible 3D (X3D) scene graph description of dynamically generated nodes and links that define a dynamic network structure. The directed acyclic graph (DAG) of X3D objects, which includes an internal ECMAScript CDATA text block in the root <Script> node, is discussed. The scene access interface is for the Vivaty Player and uses the Browser createX3DFromString() function to generate X3D objects. Network nodes can be interactively manipulated, and the appropriate links are extruded if/when these nodes are repositioned. In addition, the X3D design provides for (1) exploitation of asynchronous Javascript and extensible markup language technology for X3D (AjaX3D) in rapid development of complex networks and (2) animation of packet exchange between network nodes connected by a link through dynamic update of the cycleInterval attribute in a <TimeSensor> node. The intent is to discuss the DAG for a particular X3D scene and then provide actual code to assist the user in tailoring it to his/her particular situation.

15. SUBJECT TERMS
X3D, SAI, ECMAScript, dynamic network, scene graph

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Andrew M. Neiderer
Unclassified	Unclassified	Unclassified	UU	62	19b. TELEPHONE NUMBER (*Include area code*) 410-278-3203

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

Acknowledgments

In addition to advice from the Extensible three-dimensional (X3D) public e-mail list (http://www.web3d.org/ x3d/publiclists/), the author specifically wishes to acknowledge the insight provided from Sergey Bederov, Parallel Graphics, Inc. His suggestions made an X3D visualization of dynamic networks possible.

INTENTIONALLY LEFT BLANK.

1. Introduction

A network description and presentation of intelligence data can assist in analyzing interrelated groups or individuals. Valdis Krebs, a leading authority on social network analysis, describes identification of a terrorist network as "an art and science of finding the important connections in a seemingly impenetrable mass of data." A node in a network could be a terrorist with a link representing communication between two people. The appropriate data that is organized and then analyzed in a timely manner in this context certainly increases the probability of developing an effective strategy for deterring a possible covert activity. This report addresses the organization of data in three-dimensional (3-D) space by introducing a computer program developed at the U.S. Army Research Laboratory called d^{2}NetVis: dynamic generation of nodes and links that define a dynamic network structure using the Extensible 3D (X3D) application programming interface (API). The extension into 3-D should help to discover hidden patterns that are often difficult to identify in two dimensions. In addition, an X3D description also supports a time-based presentation via its route mechanism.

X3D is an International Standards Organization (ISO) specification[1] that is extensible markup language-based for presenting 3-D computer graphics across the Web and allows for real-time, interactive communication with the displayed content. Five different profiles for the 28 components are defined. A component is a functionality set of X3D objects; a profile consists of components. There are nearly 170 different primitive X3D objects, or nodes, in addition to the possibility of limitless prototypes that the user can define. The functionality of each node is documented in the aforementioned specification. X3D is also open sourced and can be viewed using an X3D browser. For this report, the Vivaty Player (VP) from Vivaty, Inc.[2] was selected for viewing on a Windows machine.

Every X3D browser must support ECMAScript* bindings for exposing and accessing data within a scene. The ECMAScript program code is embedded in an X3D <Script> and allows for generation of both network nodes and dynamic links from an internal CDATA text block. Authoring these involves using the createX3DFromString() method of the X3D Browser class, as each is added to the scene. Note that Java language bindings exist for browsing in the Yumetech, Inc. Xj3D 2.0 viewer and is being considered as mentioned in section 4.

[1] ISO/IEC 19775:2004 Extensible 3D Standard. http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/ (accessed 12 October 2008).

[2] Vivaty, Inc. http://help.vivaty.com/ (accessed 17 December 2008).

* ECMAScript, where ECMA is the acronym for European Computer Manufacturers Association, is the standard for a client-side script language (http://www.ecmascript.org.).

d^2NetVis is written for the immersive profile that includes interactivity within the scene. The scene graph description is a directed acyclic graph (DAG) of X3D objects and relationships among these or a hierarchical parent/child structure of X3D objects with no cycling. In addition, X3D animation is supported by an event cascade across X3D sensor, interpolator, script, and transform nodes.

The next three sections examine tasks that were addressed in this first version of d^2NetVis: authoring of nodes and links, access of the nodes and links, and animation of information exchange along the links between nodes (collectively referred to as a^3). Each of the a^3 tasks is accomplished via an X3D <Script> using either ECMAScript or Java language bindings to scene content. Network nodes and links are currently represented as X3D <Sphere> and <Extrusion> geometries, respectively. However, for the case of animation within the network, use of X3D <Cylinder> instead of <Extrusion> is under consideration for better control of speed along a link, i.e., information flow between nodes. Complete source code is provided in the appendix. (Also note that although d^2NetVis addresses visualization of a specific kind of network, it can be easily modified for other applications using a three-component description.)

2. Authoring the X3D Scene

X3D nodes and attributes are defined in the ISO specification at http://www.web3d.org/x3d /specifications/ISO-IEC-19775-X3DAbstractSpecification/.[1] Brutzman and Daly[3] describe X3D as it existed in late 2007 and 3-D concepts in general. The following discussion includes only those X3D details necessary for d^2 NetVis.

Both network nodes and links are dynamically generated in the root script <Script DEF=”SCRIPT”> (note that X3D root nodes in our application are DEF’ed in all capital letters). An X3D node has properties called fields. A field includes an attribute value for storage of the basic data type, such as SFFloat for a floating-point value or SFVec3f for a vector of three floating-point values. Each field attribute name in SCRIPT is appropriately prefixed as either a node or link, e.g., <field name=”nodeColor”> for a node color or <field name=”linkColor”> for a link color. The two main field names in the root script that the user needs to be aware of are nodePositionVector and linkTopology. A value for the former provides the 3-D location. The latter consists of entries for a node—the total number of links for the node, followed by a list of indices to other network nodes starting at 1. As an example, the first network node in SCRIPT (see the appendix) has two links—2, 3—which are links from node one to the second and third nodes, respectively.

[3]Brutzman, D.; Daly, L. *X3D: Extensible 3D Graphics for Web Authors*; Morgan Kaufmann Publishers by Elsevier, Inc.: New York, 2007.

After defining fields, node block ECMAScript code was written to access values. The DAG for each network node is illustrated in the first figure (see figure 1). The main difference among network nodes is the location of the geometry, which is a <Sphere>. Both a <PlaneSensor> and a <TouchSensor> are attached to each network node. The <PlaneSensor> provides for movement of the node in 3-space when routed to a <Transform>. Note that <ROUTE> is not an X3D node but a mechanism that makes an abstract connection between an X3D node sending an event and the X3D node receiving that event. The first branch within the DAG of a network node is defined for the geometry and a second branch for the text.

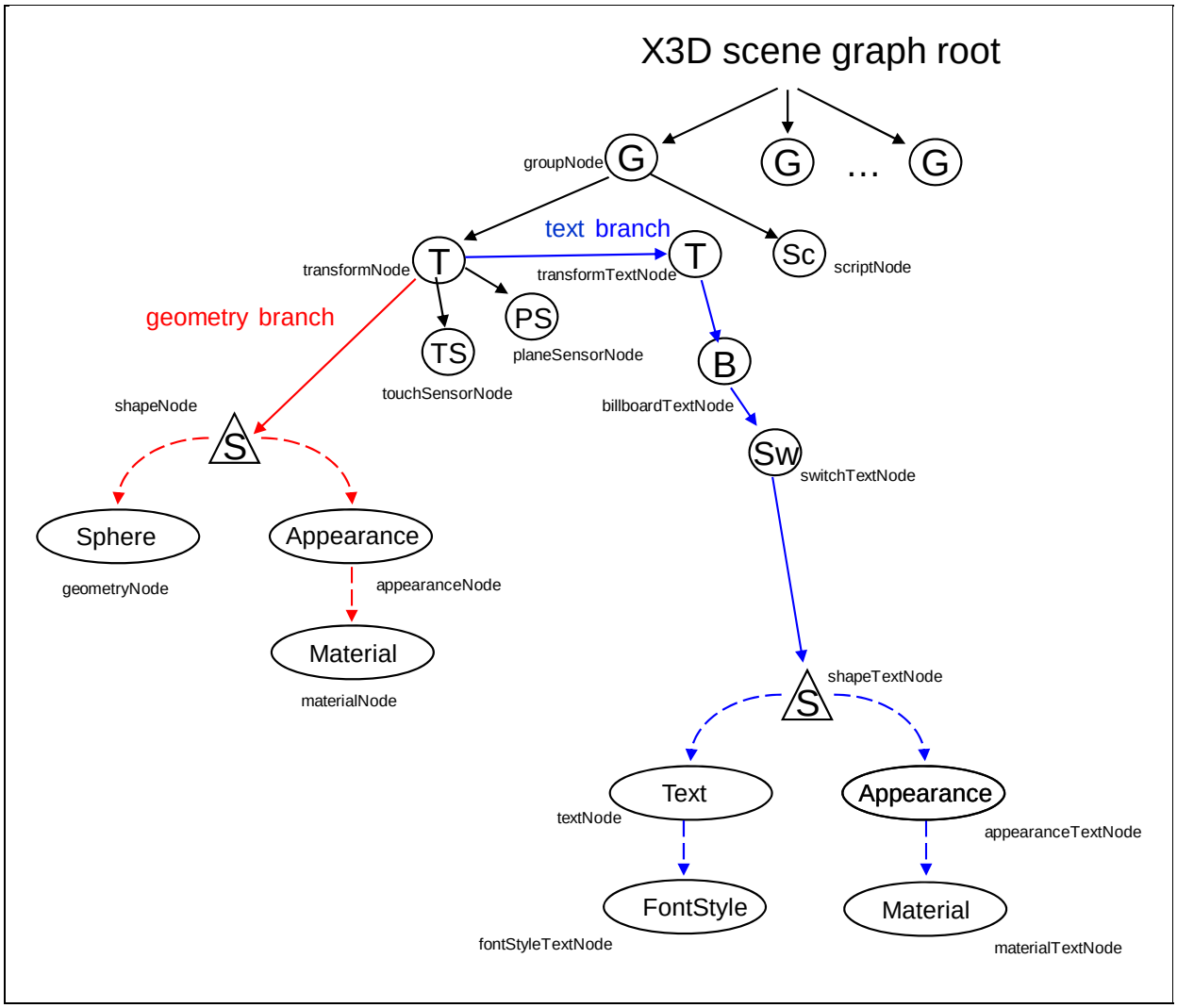

Figure 1. The DAG of X3D scene graph objects description of network nodes (includes geometric and text branches).

Animation of text on a network node, toggling in this case, is possible by simply clicking on it. The touchTime field (accessType="outputOnly") of the network node <TouchSensor> is routed to the SCRIPT_networkNode function toggleNetworkNode() (accessType="inputOnly"). (See figure 2 for an example rendering of 19-nodes representative of the 9/11 terrorist network.)

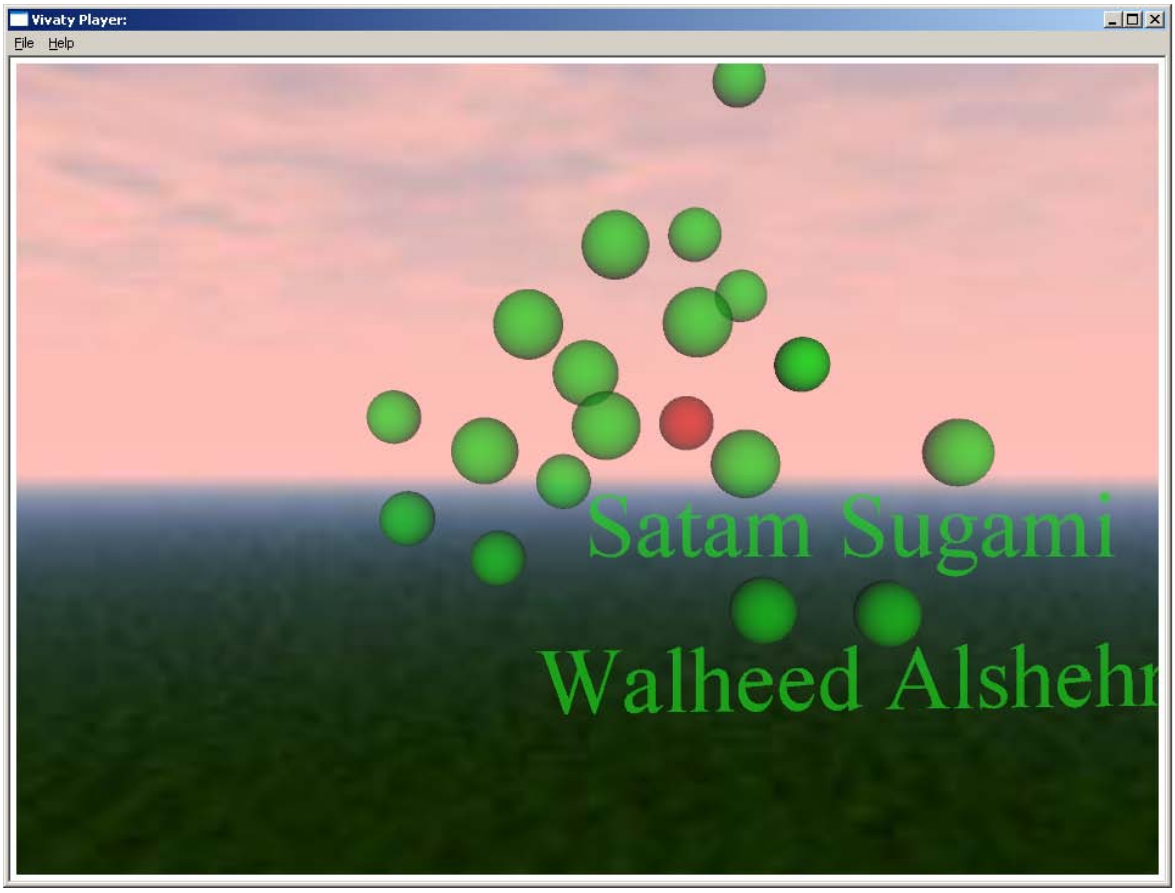

Figure 2. A 19-node network where each node is an X3D <Sphere>. The red node (Mohamed Atta) is of particular interest for this example.

A static link block is quite similar to the node block. A <TouchSensor> is attached to each link for toggling text (see DAG in figure 3), where the geometry is now an X3D <Cylinder>. But since movement in 3-space is not necessary for this geometry, an event chain is not needed, thus, no <PlaneSensor>. The total number of links between nodes that can be displayed is given by the statistical combination ${}_{n}C_{2}$ = n! / [(n – 2)! 2!]. A rendering can quickly become quite cluttered so the user must be selective when choosing which geodesic paths to display. In figure 4, links have been added to the node network example.

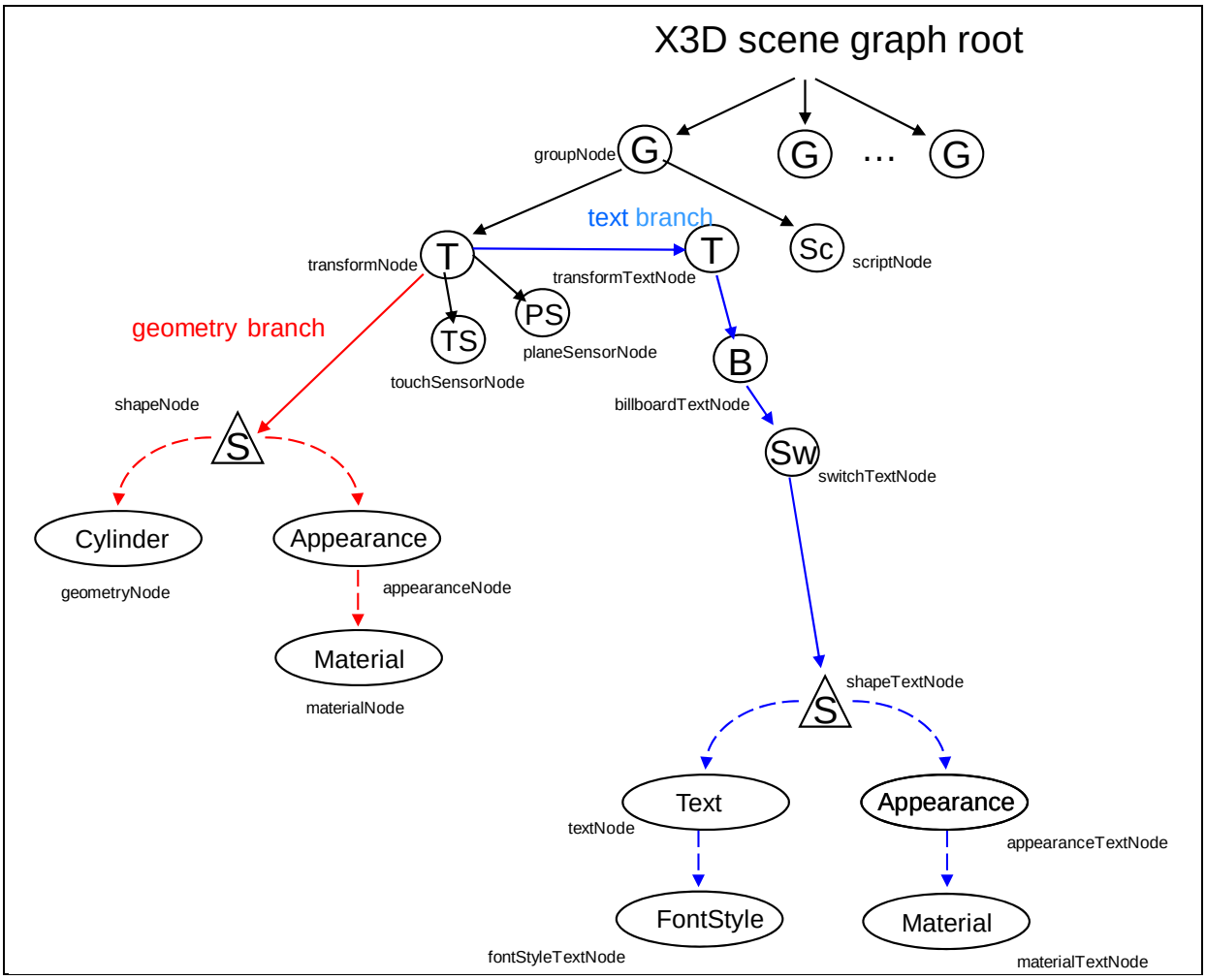

Figure 3. The DAG of X3D scene graph objects describing links between nodes for the network (including geometric and text branches).

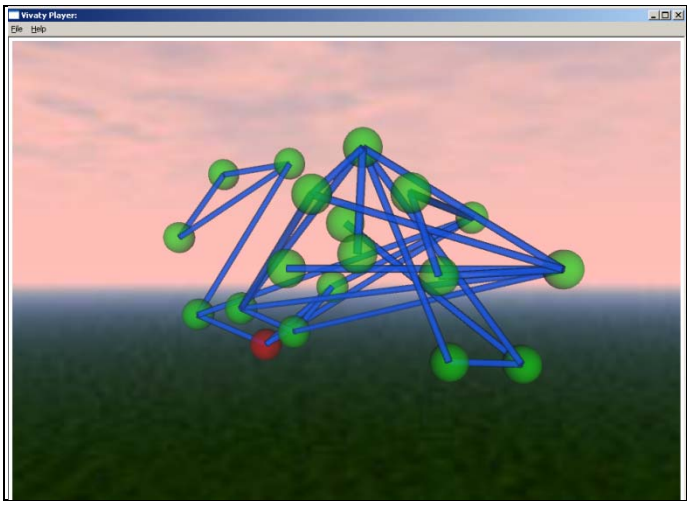

Figure 4. A 19-node, 27-link network with an X3D <TouchSensor> attached to each for toggle of text.

3. Accessing the X3D Scene for Link Extrusion

As mentioned, the DAG for static links is quite similar to network nodes (see previous section). Each link is a <Cylinder> instead of a <Sphere>. But to ensure the integrity of the network for an immersive environment, we now make all links dynamic.

In the dynamic link block of d^2 NetVis, each link is now an <Extrusion>. The spine_changed field (accessType="outputOnly") of the SCRIPT_link node is routed to the MFVec3f set_spine field (accessType="initializeOnly") of an EXTRUSION_link (see code in the appendix). There are three functions in SCRIPT_link to ensure spine data is current: translationA(), translationZ(), and processSpine(). The implementation of these three functions is quite simple but necessary. The description of the <Extrusion> node in the specification completely defines the requirements and capabilities.

Once all necessary field values and routes are provided for each node in the scene, an interactive scene for the network exists. Figure 5 illustrates a random manipulation of the 19-node, 27-link network with dynamic links.

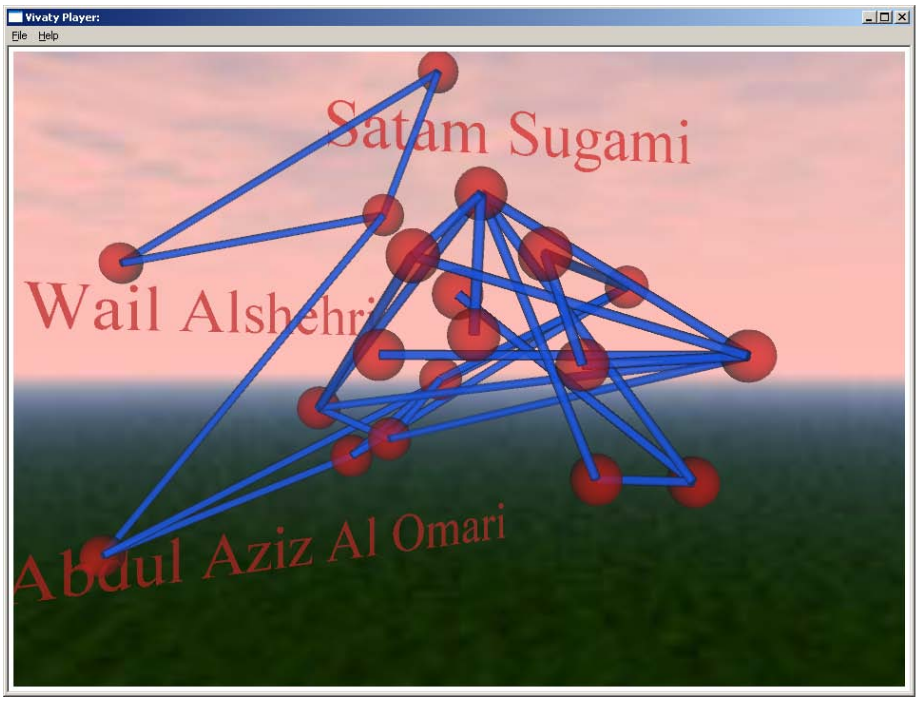

Figure 5. A 19-node, 27-link network that is dynamic for an interactive environment. Each node has an X3D <PlaneSensor> attached for movement in 3-space, and links are extruded to maintain network integrity.

4. Future Efforts

Animating information flow for a three-node subnetwork has been demonstrated (see figure 6) using the Xj3D 2.0 browser (as opposed to VP). (See http://www.yumetech.com/ for comparison of an ECMAScript scene access interface to Java.[4]) Xj3D supports both types of a script node, ECMAScript and Java, but VP was chosen in May 2008 for d^{2}NetVis due to a problem with Xj3D's handling of double quotes in CDATA text blocks; double quotes inside CDATA text blocks were necessary in this first version of d^{2}NetVis. This limitation will be revisited with possible modifications.

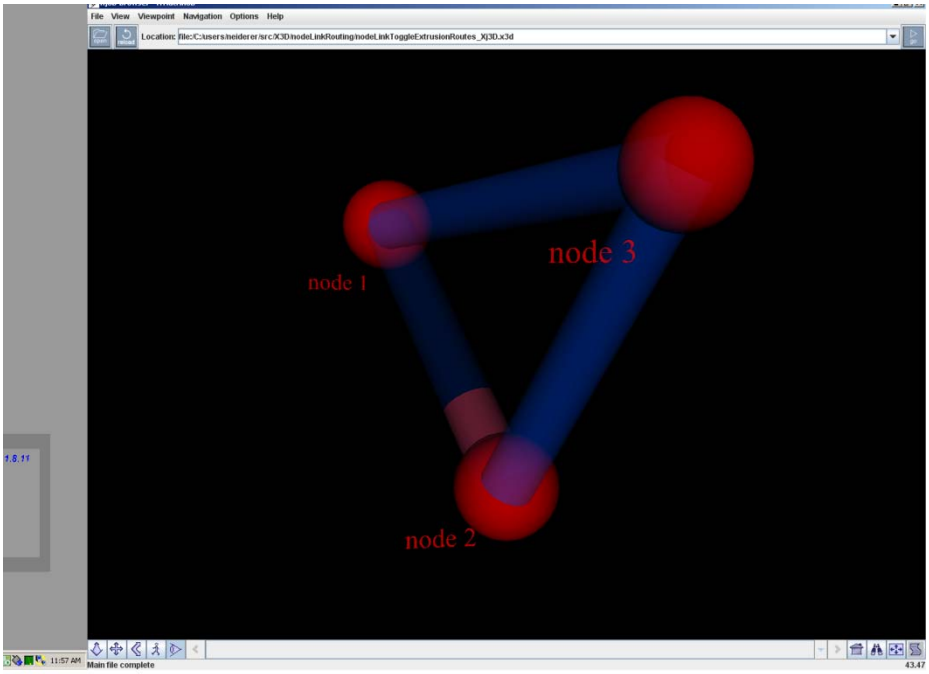

Figure 6. A three-node network illustrating animation of information exchange. Text for nodes and links can be toggled.

Animation was then defined and developed for a bilinear link (see figure 7). Speed of such exchanges can be controlled using the cycleInterval attribute of the X3D <TimeSensor>, which is routed to a <PositionInterpolator2D> and then a <TextureTransform>. The code for these two examples, where differences from d^{2}NetVis reduce to the problem of event routing of fields to appropriate X3D nodes, is not included in this report pending further research of what will trigger and drive the simulation.

[4]Yumetech. http://www.yumetech.com/ (accessed 7 January 2009).

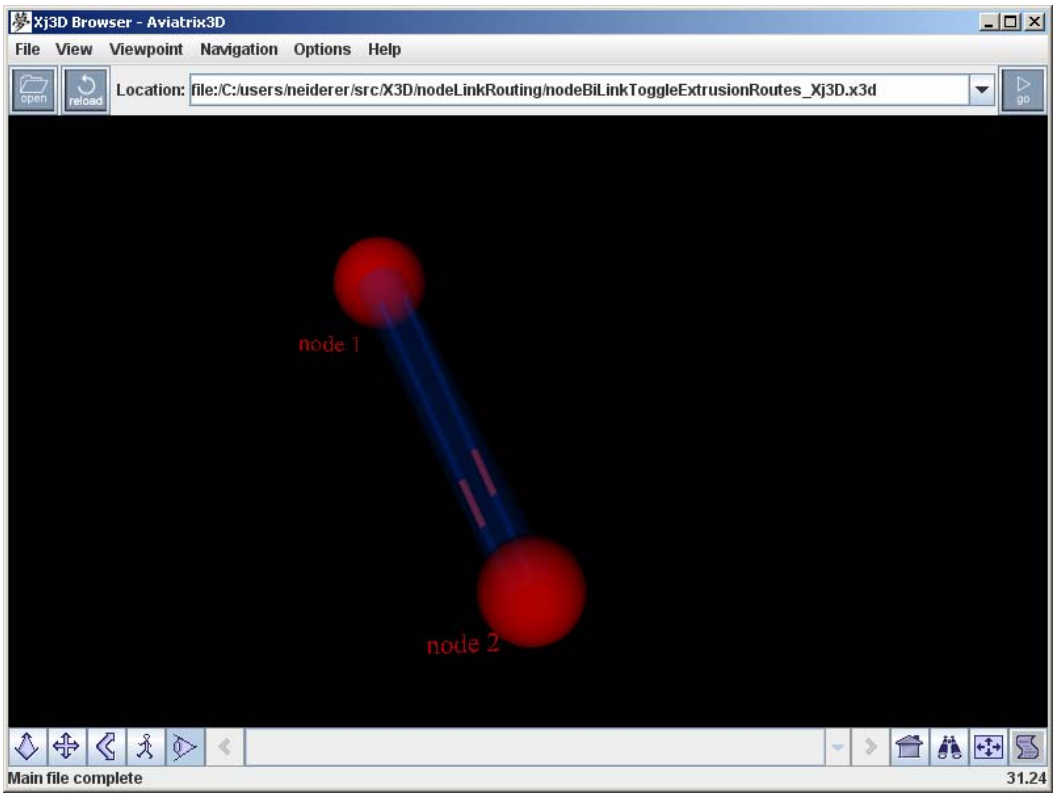

Figure 7. A bidirectional link between nodes allows for animation of information exchange. Text for both nodes and links can be toggled.

Initial development of d^{2}NetVis concentrated on the display of networks. We now plan to include logic for strategic node placement, minimizing edge length and crossings within a network, and enhance text display. Color and texture animation of text has been done in a previous X3D applications.[5] This was accomplished by first placing a <TouchSensor> on the geometry and routing the "touchTime" field to the <TimeSensor > field "startTime"; the cycleInterval attribute of this X3D node was set to some value for texture fade. The "fraction_changed" field of this <TimeSensor> was then routed to a <ScalarInterpolator> field "set_fraction." This finally sent the field "value_changed" to the appropriate <Material>. The event cascade for texture animation is documented in ARL-MR-691[5] and will be considered in a future release of d^{2}NetVis.

[5]Neiderer, A. *Visualization of a Text Network Structure Using X3D*; ARL-MR-691; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, May 2008.

5. Conclusion

The preceding discussion introduced the U.S. Army Research Laboratory's d^2NetVis tool using the X3D API for describing a network that can be subsequently visualized in an X3D viewer. Both nodes and links for the network were generated dynamically in a <Script> node using the ECMAScript language bindings for X3D. The program was written for the immersive profile and thus is interactive as well.

When viewing the X3D, the user is able to manipulate the scene for any particular orientation and can investigate possible relationships within the data that may be difficult to discern in two-dimensional space.

As mentioned in the previous section, research and implementation of animation, both positional and texture, is being considered.

INTENTIONALLY LEFT BLANK.

Appendix. The X3D and ECMAScript for d^2 NetVis

The following extensible three-dimensional (X3D) application, including the root <Script>, is for building a dynamic network structure of nodes and links. The links are defined by an extrusion geometry to allow for interactive manipulation of nodes while maintaining the integrity of the network. The network consists of 19 nodes and 27 links.

Note that X3D field names and ECMAScript variable names are camel-cased, i.e., begin with a lower-case letter. Successive words are then capitalized with no spacing.

The source code in this appendix appears in its original form, without editorial change.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN"
                "http://www.web3d.org/specifications/x3d-3.1.dtd">

<X3D profile="Immersive" version="3.1">
  <head>
    <meta name="filename"    content="create911NetworkDL_VP.x3d"/>

    <meta name="description" content="dynamically build a dynamic (d^2)
                              network of nodes and extruded bi-links
                              from an internal ECMAScript CDATA
                              text block of an X3D script node.

                             Use Vivaty Player to view the X3D."/>

    <meta name="author"      content="Andrew M. Neiderer,
                              US Army Research Laboratory."/>

    <meta name="date"        content="9 December 2008."/>

    <meta name="reference"   content="Sergey Bederov,
                              Parallel Graphics, Inc."/>
  </head>

  <Scene>
    <!-- need this for billboard bug in VP -->

    <Transform translation="0.0 0.0 -9.9">
      <Shape DEF="SHAPE_node">
          <Appearance>
            <Material/>
          </Appearance>
```

```
      <Sphere radius="0.001"/>
  </Shape>
</Transform>

<Background groundAngle="0.1, 1.309, 1.571"
          groundColor="0.0 0.0 0.0, 0.0 0.1 0.3, 0.0 0.2 0.5, 0.0 0.3 0.8"
          skyAngle="0.1, 0.15, 1.309, 1.571"
          skyColor="0.4 0.4 0.1, 0.4 0.4 0.1, 0 0.1 0.3, 0 0.2 0.6, 0.8 0.8 0.8"
          frontUrl="horizon_1_front.jpg"
          backUrl="horizon_1_back.jpg"
          leftUrl="horizon_1_left.jpg"
          rightUrl="horizon_1_right.jpg"
          topUrl="horizon_1_top.jpg"
          bottomUrl="horizon_1_bottom.jpg"/>

<Fog DEF="FOG"
    color="1.0 1.0 1.0" fogType="LINEAR" visibilityRange="75.0"/>

<!-- include Ajax technology -->

<Script DEF="SCRIPT_Ajax"
       directOutput="true">
  <![CDATA[ecmascript:
                function initialize()
                {
                  Browser.print(' ');
                  Browser.print('  SCRIPT_Ajax initialize()');
                }
         ]]>
</Script>

<!-- this root is parent of generated nodes and links -->
<Group DEF="GROUP"/>
```

```
<!-- the root Script for generating nodes and links of network -->

<Script DEF="SCRIPT">
  <field name="nodeColor"          type=      "SFColor"
                          value=     "0.8 0.0 0.0"
                          accessType="initializeOnly"/>
  <field name="nodeColorR"         type=      "SFColor"
                          value=     "0.8 0.0 0.0"
                          accessType="initializeOnly"/>
  <field name="nodeColorG"         type=      "SFColor"
                          value=     "0.0 0.8 0.0"
                          accessType="initializeOnly"/>
  <field name="nodeColorB"         type=      "SFColor"
                          value=     "0.0 0.0 0.8"
                          accessType="initializeOnly"/>
  <field name="nodeTransparency"   type=      "SFFloat"
                          value=     "0.3"
                          accessType="initializeOnly"/>
  <field name="nodeRadius"         type=      "SFFloat"
                          value=     "0.25"
                          accessType="initializeOnly"/>
  <field name="nodeRadiusPrimary"  type=      "SFFloat"
                          value=     "0.4"
                          accessType="initializeOnly"/>
  <field name="nodeRadiusSecondary" type=     "SFFloat"
                          value=     "0.25"
                          accessType="initializeOnly"/>
  <field name="nodeRadiusTertiary"  type=     "SFFloat"
                          value=     "0.1"
                          accessType="initializeOnly"/>

  <!-- Cartesian coordinates of node 1, node 2, ... node 19.  -->
```

```
<!--                                        -->
<!-- Note the delimiter -999.99999 etc is necessary :-(     -->

<field name="nodePositionVector"  type=      "MFVec3f"
                          value=     "
                                    -2.01240    1.71360    0.60000,
                                    -2.71140    0.67860    0.70000,
                                    -0.91680    1.85220    0.80000,
                                    -2.36220   -0.56100    0.90000,
                                    -1.26480   -1.01280    1.00000,
                                    -0.22200   -0.10740    1.10000,
                                    -0.80580   -0.79560    1.20000,
                                    -1.63560   -0.44460    1.10000,
                                    -0.42720    1.03740    3.00000,
                                     0.13140    0.31920    2.90000,
                                     0.20040    1.65120    2.80000,
                                     1.29960   -1.04640    2.70000,
                                     2.25300   -1.08660    2.60000,
                                    -0.01920    0.72840    2.60000,
                                    -0.76680    0.13980    2.70000,
                                     2.71140    0.12660    2.80000,
                                     1.13700    0.04260    2.90000,
                                     0.78060    1.05240    3.00000,
                                     1.93860    0.95520    1.10000,
                                   -999.99999 -999.99999 -999.99999
                                  "
                          accessType="initializeOnly"/>

<!-- text on node 1, node 2, ... node 19; parallel -->
<!--  to "nodePositionVector".                    -->

<field name="nodeName"           type=      "MFString"
                          value=     "
```

```
                'Satam Sugami',
                'Wail Alshehri',
                'Walheed Alshehri',
                'Abdul Aziz Al Omari',
                'Mohamed Atta',
                'Ziad Jarrah',
                'Ahmed Al Haznawi',
                'Saeed Alghamdi',
                'Ahmed Alnami',
                'Salem Alhazmi',
                'Nawaf Alhazmi',
                'Khalid Al Mihdhar',
                'Hani Hanjour',
                'Majed Moqed',
                'Ahmed Alghamdi',
                'Hamza Alghamdi',
                'Mohand Alshehri',
                'Fayez Ahmed',
                'Marwan Al Shehhi',
                'infinity'
               "
        accessType="initializeOnly"/>
<field name="nodeIsMemberOf"  type=  "MFString"
        value=  "
                'Mashhadan cell',
                'al-Qaeda',
                'c cell',
                'd cell',
                'e cell',
                'f cell',
                'g cell',
                'h cell',
                'i cell',
```

```
                                    'j cell',
                                    'k cell',
                                    'l cell',
                                    'm cell',
                                    'n cell',
                                    'o cell',
                                    'p cell',
                                    'q cell',
                                    'r cell',
                                    's cell',
                                    'delimiter'
                                   "
                            accessType="initializeOnly"/>

<field name="linkColor"          type=      "SFColor"
                            value=     "0.0 0.3 1.0"
                            accessType="initializeOnly"/>
<field name="linkTransparency"    type=      "SFFloat"
                            value=     "0.6"
                            accessType="initializeOnly"/>
<field name="linkRadius"         type=      "SFFloat"
                            value=     "0.15"
                            accessType="initializeOnly"/>
<field name="linkHeight"         type=      "SFFloat"
                            value=     "1.0"
                            accessType="initializeOnly"/>

<!-- link(s) from node 1, node 2, ... node 19; note that for n nodes,  -->
<!--  max no. of links = n! / [(n - 2)! 2!].                          -->
<!--                                                  -->
<!-- e.g. node 1 has "2" links:                          -->
<!--                      node 1 to node "2",                  -->
<!--                      node 1 to node "3".                  -->
```

```
<field name="linkTopology"       type=      "MFInt32"
                                 value=     "
                                            2,
                                             2,  3,
                                            2,
                                             1,  3,
                                            3,
                                             1,  2,  4,
                                            3,
                                             3,  5, 19,
                                            3,
                                             4,  6, 19,
                                            3,
                                             5,  7, 19,
                                            3,
                                             6,  8, 16.
                                            4,
                                             7,  9, 11, 16.
                                            3,
                                             8, 11, 16,
                                            1,
                                             11,
                                            6,
                                             8,  9, 10, 12, 13, 16,
                                            2,
                                             11, 13,
                                            3,
                                             11, 12, 14,
                                            1,
                                             13,
                                            1,
                                             16,
```

```
                                    6,
                                      7,  8,  9, 11, 15, 17,
                                    2,
                                     16, 18,
                                    2,
                                     17, 19,
                                    4,
                                      4,  5,  6, 18
                                  "
                          accessType="initializeOnly"/>

<!-- text on link(s); parallel to "linkTopology". -->

<field name="linkName"           type=      "MFString"
                          value=     "
                                   'link from  1 to  2',
                                   'link from  1 to  3',

                                   'link from  2 to  1',
                                   'link from  2 to  3',

                                   'link from  3 to  1',
                                   'link from  3 to  2',
                                   'link from  3 to  4',

                                   'link from  4 to  3',
                                   'link from  4 to  5',
                                   'link from  4 to 19',

                                   'link from  5 to  4',
                                   'link from  5 to  6',
                                   'link from  5 to 19'.
```

'link from 6 to 5',
'link from 6 to 7',
'link from 6 to 19',

'link from 7 to 6',
'link from 7 to 8',
'link from 7 to 16',

'link from 8 to 7',
'link from 8 to 9',
'link from 8 to 11',
'link from 8 to 16',

'link from 9 to 8',
'link from 9 to 11',
'link from 9 to 16',

'link from 10 to 11',

'link from 11 to 8',
'link from 11 to 9',
'link from 11 to 10',
'link from 11 to 12',
'link from 11 to 13',
'link from 11 to 16'

'link from 12 to 11',
'link from 12 to 13',

'link from 13 to 11',
'link from 13 to 12',
'link from 13 to 14',

```
                                    'link from 14 to 13',

                                    'link from 15 to 16',

                                    'link from 16 to  7',
                                    'link from 16 to  8',
                                    'link from 16 to  9',
                                    'link from 16 to 11',
                                    'link from 16 to 15',
                                    'link from 16 to 17'

                                    'link from 17 to 16',
                                    'link from 17 to 18',

                                    'link from 18 to 17',
                                    'link from 18 to 19',

                                    'link from 19 to  4',
                                    'link from 19 to  5',
                                    'link from 19 to  6',
                                    'link from 19 to 18'
                                   "
                            accessType="initializeOnly"/>

<field name="children_changed"    type=      "MFNode"
                            accessType="outputOnly"/>

<![CDATA[ecmascript:
                function initialize()
                {
                  var nodeColorR, nodeColorG;

                  var nodeLinks;
```

```
var offset;
var links;
var dynamicLink;

var i, j, k, l;

// node block -
//  (1) build using nodeString, then
//  (2) access in the for loop.

        var nodeString =
    '<?xml version="1.0" encoding="UTF-8"?>                                    ' +
    '<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN"                                ' +
    '                  "http://www.web3d.org/specifications/x3d-3.1.dtd">              ' +
    '                                                                ' +
    '<X3D profile="Immersive" version="3.1">                                   ' +
    '  <Scene>                                                       ' +
    '    <Group DEF="GROUP_node">                                              ' +
    '      <!-- the network node -->                                          ' +
    '                                                                ' +
    '      <Transform DEF="TRANSFORM_node"                                         ' +
    '                 translation="-2.01240 1.7136 0.6">                       ' +
    '        <!-- appearance and geometry -->                                  ' +
    '                                                                ' +
    '        <Shape DEF="SHAPE_node">                                          ' +
    '          <Appearance DEF="APPEARANCE_node">                                    ' +
    '            <Material DEF="MATERIAL_node"                                   ' +
    '                      diffuseColor="0.8 0.0 0.0"                            ' +
    '                      transparency="0.3"/>                                  ' +
    '          </Appearance>                                                 ' +
    '                                                                ' +
    '          <Sphere DEF="SPHERE_node"                                         ' +
    '                  radius="0.25"/>                                       ' +
```

```
'      </Shape>                                                    ' +
'                                                                  ' +
'      <!-- text for the network node -->                          ' +
'                                                                  ' +
'      <Transform DEF="TRANSFORM_nodeText"                                 ' +
'              translation="-1.5 -0.75 -0.5">                      ' +
'       <!-- rotating text -->                                     ' +
'                                                                  ' +
'       <Billboard DEF="BILLBOARD_nodeText">                               ' +
'        <!-- toggle display of text on the network node -->               ' +
'                                                                  ' +
'        <Switch DEF="SWITCH_networkNode">                                 ' +
'         <!-- appearance and text -->                              ' +
'                                                                  ' +
'         <Shape DEF="SHAPE_nodeText">                                     ' +
'          <Appearance DEF="APPEARANCE_nodeText">                                ' +
'           <Material USE="MATERIAL_node"/>                                 ' +
'          </Appearance>                                           ' +
'                                                                  ' +
'          <Text DEF="TEXT_node"                                           ' +
'              containerField="geometry"                           ' +
'              maxExtent="0.0">                                    ' +
'           <FontStyle DEF="FONT_STYLE_nodeAndLink"                                ' +
'                   containerField="fontStyle"                     ' +
'                   horizontal="true"                              ' +
'                   leftToRight="true"                             ' +
'                   size="1.0"                                     ' +
'                   spacing="1.0"                                  ' +
'                   style="PLAIN"                                  ' +
'                   topToBottom="true"/>                           ' +
'          </Text>                                                 ' +
'         </Shape>                                                 ' +
'        </Switch>                                                 ' +
```

```
'       </Billboard>                                                    ' +
'     </Transform>                                                      ' +
'                                                                 ' +
'     <TouchSensor DEF="TOUCH_SENSOR_node"                                ' +
'               enabled="true"/>                                        ' +
'                                                                 ' +
'     <PlaneSensor DEF="PLANE_SENSOR_node"                                ' +
'               offset="-2.01240 1.71360 0.6"/>                         ' +
'   </Transform>                                                        ' +
'                                                                 ' +
'   <!-- toggle text on the network node -->                              ' +
'                                                                 ' +
'   <Script DEF="SCRIPT_networkNode"                                      ' +
'         directOutput="true">                                          ' +
'     <field name="switchText"            type="SFNode"                   ' +
'                           accessType="initializeOnly">                ' +
'       <Switch USE="SWITCH_networkNode"/>                                ' +
'     </field>                                                          ' +
'                                                                 ' +
'     <field name="toggleNetworkNode"       type= "SFTime"                ' +
'                           accessType="inputOnly"/>                    ' +
'                                                                 ' +
'     <field name="translation_textChanged" type= "SFVec3f"               ' +
'                           accessType="outputOnly"/>                   ' +
'                                                                 ' +
'     <' + '![CDATA[ecmascript:                                         ' +
'                   function initialize()                               ' +
'                   {                                                   ' +
'                     Browser.print("SCRIPT_networkNode initialize()");      ' +
'                   }                                                   ' +
'                                                                 ' +
'                   function toggleNetworkNode(dummy)                   ' +
'                   {                                                   ' +
```

```
'                          if ( switchText.whichChoice < 0 ) {                ' +
'                            switchText.whichChoice = 0;                      ' +
'                          }                                                  ' +
'                          else {                                             ' +
'                            switchText.whichChoice = -1;                     ' +
'                          }                                                  ' +
'                        }                                                    ' +
'           ]' + ']>                                                          ' +
'    </Script>                                                                  ' +
'                                                                            ' +
'    <!-- animation -->                                                          ' +
'                                                                            ' +
'    <ROUTE fromNode="TOUCH_SENSOR_node"  fromField="touchTime"                    ' +
'         toNode=  "SCRIPT_networkNode" toField=  "toggleNetworkNode"/>            ' +
'    <ROUTE fromNode="PLANE_SENSOR_node"  fromField="translation_changed"           ' +
'         toNode=  "TRANSFORM_node"     toField=  "translation"/>                 ' +
'                              <!-- text translation -->                      ' +
'    <ROUTE fromNode="SCRIPT_networkNode" fromField="translation_textChanged"       ' +
'         toNode=  "TRANSFORM_nodeText" toField=  "translation"/>                 ' +
'   </Group>                                                                   ' +
'  </Scene>                                                                    ' +
'</X3D>                                                                       ';

Browser.print(' ');
Browser.print('  root script node, ie SCRIPT,');
Browser.print('    internal CDATA ECMAScript initialize().');
Browser.print(' ');
Browser.print('    node block - ');
Browser.print(' ');

// the scene graph is built using the createX3DFromString() service
//  for each nodePositionVector, ie node; note the delimiter
//  9999.99999 ...
```

```
j = 1;

children_changed = new MFNode();

for ( i = 1; i < nodePositionVector.length; i++ ) {

  // use createX3DFromString() method of Browser class
  //  since script node, which is used to toggle text
  //  on a network node, has user-defined fields.

  var scene = Browser.createX3DFromString(nodeString);

  var nodes = scene.rootNodes;

  Browser.print('      nodes.length = ' + nodes.length);
  Browser.print(' ');

  // nodes must be removed before adding new ones

  for ( k = 0; k < nodes.length; k++ ) {
    scene.removeRootNode(nodes[k]);
  }

  // the Group node holds
  //  Transform node (groupNode.children[0]) and
  //  Script node (groupNode.children[1]).

  var groupNode = nodes[0];

  var defGroupNode = '      GROUP_node' + i;

  Browser.print(defGroupNode);
```

```
// the Transform node holds
//  Shape node (transformNode.children[0]),
//  text Transform node (transformNode.children[1]),
//  TouchSensor node (transformNode.children[2]), and
//  PlaneSensor node (transformNode.children[3]);
//
// parent is groupNode.

var transformNode = groupNode.children[0];

var defTransformNode = '        TRANSFORM_node' + i;

transformNode.translation.x = nodePositionVector[i - 1].x;
transformNode.translation.y = nodePositionVector[i - 1].y;
transformNode.translation.z = nodePositionVector[i - 1].z;

Browser.print(defTransformNode);
Browser.print('          translation = ' + nodePositionVector[i - 1].x + ' ' +
                                nodePositionVector[i - 1].y + ' ' +
                                nodePositionVector[i - 1].z);
Browser.print(' ');

// the Shape node holds
//  appearance node (shapeNode.appearance) and
//  geometry node (shapeNode.geometry);
//
// parent is transformNode.

var shapeNode = transformNode.children[0];

var defShapeNode = '          SHAPE_node' + i;
```

```
                Browser.print(defShapeNode);

                // the Appearance node holds
                //  material node (appearanceNode.material);
                //
                // parent is shapeNode.

                var appearanceNode = shapeNode.appearance;

                var defAppearanceNode = '           APPEARANCE_node' + i;

                Browser.print(defAppearanceNode);

                // Material node is a leaf node;
                //
                // parent is appearanceNode.

                var materialNode = appearanceNode.material;

                var defMaterialNode = '             MATERIAL_node' + i;

//              if ( i == 3 ) {
                  // primary HVI

//                  nodeColorR = new SFColor(0.8,0.0,0.3);
//
//                  materialNode.diffuseColor = nodeColorR;
                  materialNode.diffuseColor.r = 0.8;
                  materialNode.diffuseColor.g = 0.0;
                  materialNode.diffuseColor.b = 0.3;
//                }
//                else {
                  // secondary HVI
```

```
//
//                  nodeColorG = new SFColor(0.0,0.8,0.3);
//
//                  materialNode.diffuseColor[0] = nodeColorG[0];
//                  materialNode.diffuseColor[1] = nodeColorG[1];
//                  materialNode.diffuseColor[2] = nodeColorG[2];

//                  materialNode.diffuseColor.r = 0.0;
//                  materialNode.diffuseColor.g = 0.8;
//                  materialNode.diffuseColor.b = 0.3;
//                }
                  materialNode.transparency = nodeTransparency;

                  Browser.print(defMaterialNode);
                  Browser.print('            diffuseColor = ' + materialNode.diffuseColor.r + ' ' +
                                                materialNode.diffuseColor.g + ' ' +
                                                materialNode.diffuseColor.b);
                  Browser.print('            transparency = ' + materialNode.transparency);

                  // geometry node is a Sphere which is a leaf node;
                  //
                  // parent is shapeNode.

                  var geometryNode = shapeNode.geometry;

                  var defGeometryNode = '          GEOMETRY_node' + i;

                  geometryNode.radius = nodeRadius;

                  Browser.print(defGeometryNode);
                  Browser.print('            sphere radius = ' +
                            geometryNode.radius);
                  Browser.print(' ');
```

```
// text Transform node holds
//  text Billboard node (transformText.children[0]);
//
// parent is transformNode.

var transformText = transformNode.children[1];

var defTransformText = '          TRANSFORM_text' + i;

Browser.print(defTransformText);

// text Billboard node for rotating text on the node and holds
//  Switch node;
//
// parent is transformText.

var billboardText = transformText.children[0];

var defBillboardText = '          BILLBOARD_text' + i;

Browser.print(defBillboardText);

// text Switch node to toggle text on the node;
//
// parent is billboardText.

var switchText = billboardText.children[0];

var defSwitchText = '             SWITCH_text' + i;

// initial display of first 4 nodes
```

```
if ( i < 4 )
  switchText.whichChoice =  0;
else
  switchText.whichChoice = -1;

Browser.print(defSwitchText);
Browser.print('               whichChoice = ' + switchText.whichChoice);
Browser.print(' ');

// text Shape node holds
//  appearance node (shapeText.appearance) and
//  text node (shapeText.geometry);
//
// parent is switchText.

var shapeText = switchText.children[0];

var defShapeText = '               SHAPE_text' + i;

Browser.print(defShapeText);

// text Appearance node holds
//  material node (appearanceText.material);
//
// parent is shapeText.

var appearanceText = shapeText.appearance;

var defAppearanceText = '                APPEARANCE_text' + i;

Browser.print(defAppearanceText);

// text Material node is a leaf node;
```

```
//
// parent is appearanceText.

var materialText = appearanceText.material;

var defMaterialText = '                MATERIAL_text' + i;

materialText.diffuseColor[0] = nodeColorR[0];
materialText.diffuseColor[1] = nodeColorR[1];
materialText.diffuseColor[2] = nodeColorR[2];

materialText.transparency = nodeTransparency;

Browser.print(defMaterialText);

// Text node holds
//  text FontStyle node (textNode.fontStyle);
//
// parent is shapeText

var textNode = shapeText.geometry;

var defTextNode = '                TEXT_node' + i;

var textLength = nodeName[i - 1].length;

var text = new String(textLength);

text = nodeName[i - 1];

textNode.string[0] = text;

Browser.print(defTextNode);
```

```
Browser.print('                textLength = ' + textLength);
Browser.print('                text = ' + text);
Browser.print('                maxExtent = ' + textNode.maxExtent);
Browser.print(' ');

// text FontStyle node is a leaf node;
//
// parent is textNode.

var fontStyleText = textNode.fontStyle;

var defFontStyleText = '                FONT_STYLE_text' + i;

//              fontStyleText.containerField = "fontStyle";
//              fontStyleText.family = "SERIF";
                fontStyleText.horizontal = true;
//              fontStyleText.justify = "BEGIN";
fontStyleText.leftToRight = true;
fontStyleText.size = 1.0;
fontStyleText.spacing = 1.0;
fontStyleText.style = "PLAIN";
fontStyleText.topToBottom = true;

Browser.print(defFontStyleText);
Browser.print('                horizontal = ' + fontStyleText.horizontal);
Browser.print('                leftToRight = ' + fontStyleText.leftToRight);
Browser.print('                size = ' + fontStyleText.size);
Browser.print('                spacing = ' + fontStyleText.spacing);
Browser.print('                style = ' + fontStyleText.style);
Browser.print('                topToBottom = ' + fontStyleText.topToBottom);
Browser.print(' ');

// TouchSensor node is a leaf node;
```

```
//
// parent is transformNode.

var touchSensorNode = transformNode.children[2];

var defTouchSensorNode = '          TOUCH_SENSOR_node' + i;

touchSensorNode.enabled = "true";

Browser.print(defTouchSensorNode);
Browser.print(' ');

// PlaneSensor node is a leaf node;
//
// parent is transformNode.

var planeSensorNode = transformNode.children[3];

var defPlaneSensorNode = '          PLANE_SENSOR_node' + i;

planeSensorNode.offset.x = nodePositionVector[i - 1].x;
planeSensorNode.offset.y = nodePositionVector[i - 1].y;
planeSensorNode.offset.z = nodePositionVector[i - 1].z;

Browser.print(defPlaneSensorNode);
Browser.print('            offset = ' + nodePositionVector[i - 1].x + ' ' +
                              nodePositionVector[i - 1].y + ' ' +
                              nodePositionVector[i - 1].z);

// Script node is a leaf node;
//
// parent is groupNode.
```

```
  var scriptNode = groupNode.children[1];

  var defScriptNode = '        SCRIPT_node' + i;

  Browser.print(defScriptNode);
  Browser.print(' ');

  // update scene graph
  children_changed[children_changed.length] = groupNode;
}

// dynamic link block -
//  (1) build using dynamicLinkString, then
//  (2) access in the for loop.

        var dynamicLinkString =
    '<?xml version="1.0" encoding="UTF-8"?>                                    ' +
    '<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN"                                ' +
    '                  "http://www.web3d.org/specifications/x3d-3.1.dtd">              ' +
    '                                                                ' +
    '<X3D profile="Immersive" version="3.1">                                   ' +
    '  <Scene>                                                          ' +
    '    <Group DEF="GROUP_link">                                                ' +
    '      <!-- the link -->                                                ' +
    '                                                                ' +
    '       <Shape DEF="SHAPE_link">                                           ' +
    '         <Appearance DEF="APPEARANCE_link">                                       ' +
    '           <Material DEF="MATERIAL_link"                                    ' +
    '                     diffuseColor="0.0 0.3 1.0"                             ' +
    '                     transparency="0.6"/>                                   ' +
    '         </Appearance>                                                  ' +
    '                                                                ' +
    '         <Extrusion DEF="EXTRUSION_link"                                  ' +
```

```
'                crossSection=" 0.025  0.05,                                  ' +
'                               0.05   0.0,                                 ' +
'                               0.025 -0.05,                                 ' +
'                              -0.025 -0.05,                                 ' +
'                              -0.05   0.0,                                 ' +
'                              -0.025  0.05,                                 ' +
'                               0.025  0.05"                                 ' +
'               solid="false"                                                ' +
'               creaseAngle="1.2"                                              ' +
'               spine="                                                      ' +
'                    -2.01240  1.71360  0.60000,                               ' +
'                    -2.71140  0.67860  0.70000"/>                               ' +
'      </Shape>                                                                ' +
'                                                                            ' +
'      <!-- text for the link -->                                                ' +
'                                                                            ' +
'      <Transform DEF="TRANSFORM_linkText"                                          ' +
'               translation="-1.5 -0.75 -0.5">                                 ' +
'        <!-- rotating text -->                                                ' +
'                                                                            ' +
'        <Billboard DEF="BILLBOARD_linkText">                                      ' +
'          <!-- toggle display of text on the link -->                           ' +
'                                                                            ' +
'          <Switch DEF="SWITCH_link">                                            ' +
'            <!-- appearance and text -->                                       ' +
'                                                                            ' +
'            <Shape DEF="SHAPE_linkText">                                        ' +
'              <Appearance DEF="APPEARANCE_linkText">                                ' +
'                <Material USE="MATERIAL_link"/>                                 ' +
'              </Appearance>                                                   ' +
'                                                                            ' +
'              <Text DEF="TEXT_link"                                           ' +
'                   containerField="geometry"                                  ' +
```

```
'                maxExtent="0.0">                                          ' +
'              <FontStyle DEF="FONT_STYLE_nodeAndLink"                          ' +
'                    containerField="fontStyle"                            ' +
'                    horizontal="true"                                    ' +
'                    leftToRight="true"                                    ' +
'                    size="1.0"                                           ' +
'                    spacing="1.0"                                         ' +
'                    style="PLAIN"                                         ' +
'                    topToBottom="true"/>                                    ' +
'            </Text>                                                      ' +
'           </Shape>                                                      ' +
'         </Switch>                                                       ' +
'       </Billboard>                                                      ' +
'     </Transform>                                                        ' +
'                                                                       ' +
'     <TouchSensor DEF="TOUCH_SENSOR_link"                                    ' +
'                enabled="true"/>                                         ' +
'                                                                       ' +
'    <!-- toggle text on the link -->                                       ' +
'                                                                       ' +
'    <Script DEF="SCRIPT_link"                                                ' +
'          directOutput="true">                                          ' +
'      <field name="switchText" type=      "SFNode"                            ' +
'                        accessType="initializeOnly">                      ' +
'        <Switch USE="SWITCH_link"/>                                         ' +
'      </field>                                                          ' +
'                                                                       ' +
'      <field name="toggleLink"         type=      "SFTime"                    ' +
'                                accessType="inputOnly"/>                  ' +
'                                                                       ' +
'      <field name="translationA"       type=      "SFVec3f"                   ' +
'                                accessType="inputOnly"/>                  ' +
'      <field name="translationZ"       type=      "SFVec3f"                   ' +
```

```
'                              accessType="inputOnly"/>                    ' +
'                                                                          ' +
'      <field name="spine"              type=      "MFVec3f"                 ' +
'                              value =    "-2.01240 1.71360 0.60000,       ' +
'                                        -2.71140 0.67860 0.70000"         ' +
'                              accessType="initializeOnly"/>               ' +
'                                                                          ' +
'      <field name="spine_changed"      type=      "MFVec3f"                 ' +
'                              accessType="outputOnly"/>                   ' +
'                                                                          ' +
'      <field name="translation_textChanged"                               ' +
'                              type= "SFVec3f"                             ' +
'                              accessType="outputOnly"/>                   ' +
'                                                                          ' +
'      <' + '![CDATA[ecmascript:                                           ' +
'                          function initialize()                           ' +
'                          {                                               ' +
'                            Browser.print("SCRIPT_link initialize()");    ' +
'                          }                                               ' +
'                                                                          ' +
'                          function toggleLink(dummy)                      ' +
'                          {                                               ' +
'                           if ( switchText.whichChoice < 0 ) {            ' +
'                             switchText.whichChoice = 0;                  ' +
'                           }                                              ' +
'                           else {                                         ' +
'                             switchText.whichChoice = -1;                 ' +
'                           }                                              ' +
'                          }                                               ' +
'                                                                          ' +
'                              function translationA(value)                ' +
'                              {                                           ' +
'                                spine[0] = value;                         ' +
```

```
'                                                                 ' +
            '                        processSpine();                          ' +
         '                        }                                        ' +
            '                                                                 ' +
            '                        function translationZ(value)             ' +
            '                        {                                        ' +
            '                          spine[1] = value;                      ' +
'                                                                 ' +
            '                          processSpine();                        ' +
            '                        }                                        ' +
'                                                                 ' +
            '                                                                 ' +
            '                        function processSpine()                  ' +
            '                        {                                        ' +
            '                          spine_changed = spine;                 ' +
            '                                                                 ' +
            '                          translation_textChanged = new SFVec3f( ' +
            '                                   (spine[0].x + spine[1].x) / 2, ' +
            '                                   (spine[0].y + spine[1].y) / 2, ' +
            '                                   (spine[0].z + spine[1].z) / 2); ' +
            '                        }                                        ' +
'           ]' + ']>                                                ' +
'    </Script>                                                     ' +
'                                                                 ' +
'    <!-- animation -->                                            ' +
'                                                                 ' +
'    <ROUTE fromNode="TOUCH_SENSOR_link"      fromField="touchTime"            ' +
'         toNode=  "SCRIPT_link"            toField=  "toggleLink"/>          ' +
'                             <!-- link extrusion "out" from node --> ' +
'    <ROUTE fromNode="SCRIPT_link"            fromField="spine_changed"          ' +
'         toNode=  "EXTRUSION_link"         toField=  "set_spine"/>           ' +
'                             <!-- text translation -->          ' +
'    <ROUTE fromNode="SCRIPT_link"            fromField="translation_textChanged"   ' +
```

```
    '            toNode=  "TRANSFORM_linkText"        toField=  "translation"/>            ' +
    '    </Group>                                                                            ' +
    '  </Scene>                                                                              ' +
    '</X3D>                                                                                  ';

Browser.print(' ');
Browser.print('    dynamic link block - ');
Browser.print(' ');

// add link(s) to scene graph using the createX3DFromString() service;

dynamicLink = 0;
offset = 0;

for ( i = 1; i < nodePositionVector.length; i++ ) {
  Browser.print('      offset = ' + offset);
  Browser.print(' ');

  nodeLinks = linkTopology[offset];

  Browser.print('      node at position vector ' + i);
  Browser.print('       has ' + nodeLinks + ' link(s).');
  Browser.print(' ');

  // dynamic links for the node

  if ( nodeLinks > 0 ) {
    for ( l = 1; l <= nodeLinks; l++ ) {
      // use createX3DFromString() method of Browser class
      //  since script node, which is used to toggle text
      //  on a link, has user-defined fields.

      var scene = Browser.createX3DFromString(dynamicLinkString);
```

```
var nodes = scene.rootNodes;

Browser.print('        nodes.length = ' + nodes.length);
Browser.print(' ');

// nodes must be removed before adding new ones ...

for ( k = 0; k < nodes.length; k++ ) {
  scene.removeRootNode(nodes[k]);
}

// the Group node holds
//  link Shape node (groupLink.children[0]),
//  text Transform node (groupLink.children[1]),
//  TouchSensor node (groupLink.children[2]) and
//  Script node (groupLink.children[3]).

var groupLink = nodes[0];

var shapeLink = groupLink.children[0];

var defShapeLink = '            SHAPE_link' + (dynamicLink + l);

Browser.print(defShapeLink);

// the Appearance node holds
//  material node (appearanceLink.material);
//
// parent is shapeLink.

var appearanceLink = shapeLink.appearance;
```

```
var defAppearanceLink = '            APPEARANCE_link' + (dynamicLink + l);

Browser.print(defAppearanceLink);

// Material node is a leaf node;
//
// parent is appearanceLink.

var materialLink = appearanceLink.material;

var defMaterialLink = '              MATERIAL_link' + (dynamicLink + l);

materialLink.diffuseColor = linkColor;
materialLink.transparency = linkTransparency;

Browser.print(defMaterialLink);
Browser.print('              diffuseColor = ' + materialLink.diffuseColor.r + ' ' +
                                materialLink.diffuseColor.g + ' ' +
                                materialLink.diffuseColor.b);
Browser.print('              transparency = ' + materialLink.transparency);

// geometry node is an Extrusion that is a leaf node;
//
// parent is shapeLink.

var geometryLink = shapeLink.geometry;

var defGeometryLink = '            GEOMETRY_link' + (dynamicLink + l);

geometryLink.spine[0] = nodePositionVector[i - 1];
geometryLink.spine[1] = nodePositionVector[linkTopology[offset + l] - 1];

Browser.print(defGeometryLink);
```

```
Browser.print('            spine[0] = ' + nodePositionVector[i - 1].x + ' ' +
                               nodePositionVector[i - 1].y + ' ' +
                               nodePositionVector[i - 1].z);
Browser.print('            spine[1] = ' + nodePositionVector[linkTopology[offset + l] - 1].x + ' ' +
                               nodePositionVector[linkTopology[offset + l] - 1].y + ' ' +
                               nodePositionVector[linkTopology[offset + l] - 1].z);
Browser.print(' ');

// text Transform node holds
//  text Billboard node (transformLinkText.children[0]);
//
// parent is transformLink.

var transformLinkText = groupLink.children[1];

var defTransformLinkText = '          TRANSFORM_linkText' + (dynamicLink + l);

transformLinkText.translation.x = (geometryLink.spine[0].x + geometryLink.spine[1].x) / 2;
transformLinkText.translation.y = (geometryLink.spine[0].y + geometryLink.spine[1].y) / 2;
transformLinkText.translation.z = (geometryLink.spine[0].z + geometryLink.spine[1].z) / 2;

Browser.print(defTransformLinkText);

// text Billboard node for rotating text on the link and holds
//  Switch node;
//
// parent is transformLinkText.

var billboardLinkText = transformLinkText.children[0];

var defBillboardLinkText = '          BILLBOARD_linkText' + (dynamicLink + l);

Browser.print(defBillboardLinkText);
```

```
// text Switch node to toggle text on the link;
//
// parent is billboardLinkText.

var switchLinkText = billboardLinkText.children[0];

var defSwitchLinkText = '               SWITCH_linkText' + (dynamicLink + l);

if ( i < 4 )
  switchLinkText.whichChoice =  0;
else
  switchLinkText.whichChoice = -1;

Browser.print(defSwitchLinkText);
Browser.print('                whichChoice = ' + switchLinkText.whichChoice);
Browser.print(' ');

// text Shape node holds
//  appearance node (shapeLinkText.appearance) and
//  text node (shapeLinkText.geometry);
//
// parent is switchLinkText.

var shapeLinkText = switchLinkText.children[0];

var defShapeLinkText = '               SHAPE_linkText' + (dynamicLink + l);

Browser.print(defShapeLinkText);

// text Appearance node holds
//  material node (appearanceLinkText.material);
//
```

```
// parent is shapeLinkText.

var appearanceLinkText = shapeLinkText.appearance;

var defAppearanceLinkText = '                APPEARANCE_linkText' + (dynamicLink + l);

Browser.print(defAppearanceLinkText);

// text Material node is a leaf node;
//
// parent is appearanceLinkText.

var materialLinkText = appearanceLinkText.material;

var defMaterialLinkText = '                MATERIAL_linkText' + (dynamicLink + l);

materialLinkText.diffuseColor = linkColor;
materialLinkText.transparency = linkTransparency;

Browser.print(defMaterialLinkText);

// Text node holds
//  text FontStyle node (textLink.fontStyle);;
//
// parent is shapeLinkText

var textLink = shapeLinkText.geometry;

var defTextLink = '                TEXT_link' + (dynamicLink + l);

var textLength;

if ( offset > 0 )
```

```
  textLength = linkName[(offset - (i - 1)) + l - 1].length;
else
  textLength = linkName[l - 1].length;

var text = new String(textLength);

if ( offset > 0 )
  text = linkName[(offset - (i - 1)) + l - 1];
else
  text = linkName[(l - 1)];

textLink.string[i] = text;

Browser.print(defTextLink);
Browser.print('                textLength = ' + textLength);
Browser.print('                text = ' + text);
Browser.print('                maxExtent = ' + textLink.maxExtent);
Browser.print(' ');

// text FontStyle is a leaf node;
//
// parent is textLink.

var fontStyleLinkText = textLink.fontStyle;

var defFontStyleText = '                FONT_STYLE_linkText' + (dynamicLink + l);

//              fontStyleText.containerField = "fontStyle";
//              fontStyleText.family = "SERIF";
                fontStyleText.horizontal = true;
//              fontStyleText.justify = "BEGIN";
                fontStyleText.leftToRight = true;
                fontStyleText.size = 1.0;
```

```
fontStyleText.spacing = 1.0;
fontStyleText.style = "PLAIN";
fontStyleText.topToBottom = true;

Browser.print(defFontStyleText);
Browser.print('                horizontal = ' + fontStyleText.horizontal);
Browser.print('                leftToRight = ' + fontStyleText.leftToRight);
Browser.print('                size = ' + fontStyleText.size);
Browser.print('                spacing = ' + fontStyleText.spacing);
Browser.print('                style = ' + fontStyleText.style);
Browser.print('                topToBottom = ' + fontStyleText.topToBottom);
Browser.print(' ');

// TouchSensor node is a leaf node;
//
// parent is transformLink.

var touchSensorLink = groupLink.children[2];

var defTouchSensorLink = '        TOUCH_SENSOR_link' + (dynamicLink + l);

touchSensorLink.enabled = "true";

Browser.print(defTouchSensorLink);
Browser.print(' ');

// Script node is a leaf node;
//
// parent is groupLink.

var scriptLink = groupLink.children[3];

var defScriptLink = '       SCRIPT_link' + (dynamicLink + l);
```

```
    Browser.print(defScriptLink);
    Browser.print(' ');
    Browser.print(' ');

                // retrieve the node at the one end and add route

                var groupNode = children_changed[i - 1];
                var transformNode = groupNode.children[0];
                var planeSensorNode = transformNode.children[3];

    Browser.currentScene.addRoute(planeSensorNode,'translation_changed',
                        scriptLink,'translationA');

                // retrieve the node at the other end and add route

                var groupNode = children_changed[linkTopology[offset + l] - 1];
                var transformNode = groupNode.children[0];
                var planeSensorNode = transformNode.children[3];

    Browser.currentScene.addRoute(planeSensorNode,'translation_changed',
                        scriptLink,'translationZ');

    // initialize spine with proper values
    scriptLink.spine = geometryLink.spine;

    // update scene graph
    children_changed[children_changed.length] = groupLink;
   }
}

dynamicLink += nodeLinks;
Browser.print('     cumulative number of links in the scene = ' + dynamicLink);
```

```
                    Browser.print(' ');
                    Browser.print(' ');

                    // update for number of node links
                    offset += nodeLinks + 1;
                  }
                }
        ]]>
  </Script>

  <ROUTE fromNode="SCRIPT"    fromField="children_changed"
       toNode=  "GROUP"     toField=  "children"/>
 </Scene>
</X3D>
```

INTENTIONALLY LEFT BLANK.

List of Symbols, Abbreviations, and Acronyms

3-D	three-dimensional
a^3	author, access, and animate
API	application programming interface
AjaX3D	asynchronous JavaScript and XML for Extensible 3D
d^2NetVis	dynamically generated nodes and links for a dynamic network visualization
DAG	directed acyclic graph
ECMAScript	European Computer Manufacturers Association scripting language
ISO	International Standards Organization
VP	Vivaty Player from Vivaty, Inc.
Xj3D	Extensible 3D viewer with Java from Yumetech, Inc.
X3D	extensible three dimensional

NO. OF
COPIES ORGANIZATION

1 (PDF only) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE ALC HRR
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK PE
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF
COPIES ORGANIZATION

2 DIRECTOR
US ARMY RSRCH LAB
AMSRD ARL CI I
B BROOME
A KOTT
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

7 DIR USARL
AMSRD ARL CI CT
A BORNSTEIN
J BRAND
A NEIDERER (4 CPS)
M THOMAS

INTENTIONALLY LEFT BLANK.